Chapter 10: Files

Contents

Time required: 60 minutes

DRY

Don't **R**epeat **Y**ourself

Code Shape

Please group program code as follows.

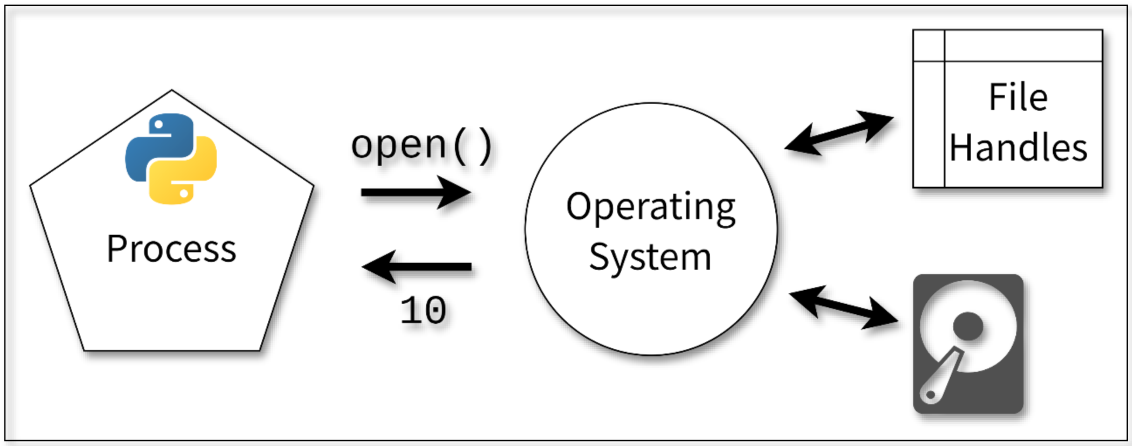
- Declare constants and variables
- Get input
- Calculate
- Display

Files

Most of the programs we have seen so far are transient in the sense that they run for a short time and produce some output. When they end, their data disappears. If you run the program again, it starts with a clean slate.

Python delegates file operations to the operating system. The operating system is the mediator between processes, such as Python, and all the system resources, such as the hard drive, RAM, and CPU time.

When you open a file with **open()**, you make a system call to the operating system to locate that file on the hard drive and prepare it for reading or writing. The operating system will then return an unsigned integer called a **file handle** on Windows and a **file descriptor** on UNIX-like systems, including Linux and macOS:



Operating systems limit the number of open files any single process can have. This number is typically in the thousands. Operating systems set this limit because if a process tries to open thousands of file descriptors, something is probably wrong with the process. Even though thousands of files may seem like a lot, it's still possible to run into the limit.

Apart from the risk of running into the limit, keeping files open leaves you vulnerable to losing data. In general, Python and the operating system work hard to protect you from data loss. But if your program or computer crashes, the usual routines may not take place, and open files can get corrupted.

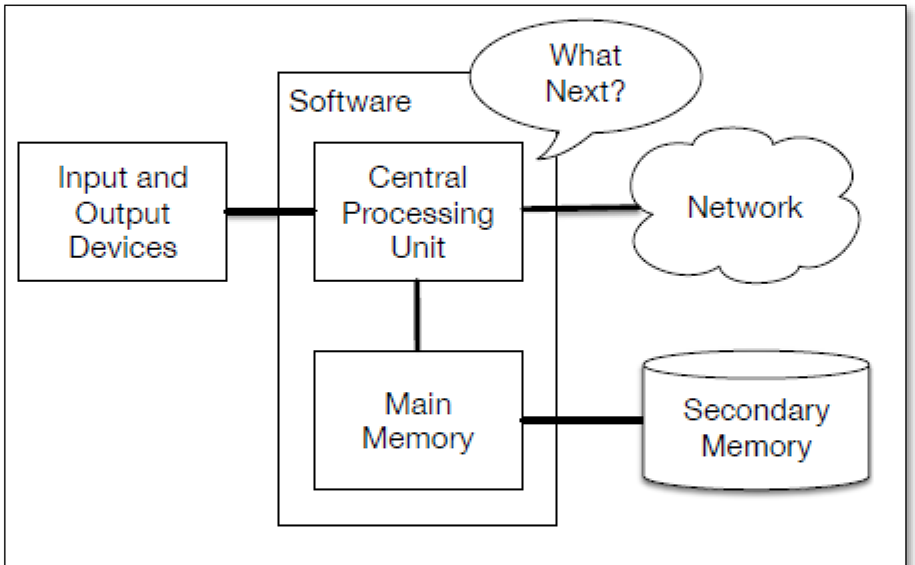


Figure 7.1: Secondary Memory

Secondary memory is not erased when the power is turned off. Or in the case of a USB flash drive, the data we write from our programs can be removed from the system and transported to another system.

We will primarily focus on reading and writing text files such as those we create in a text editor.

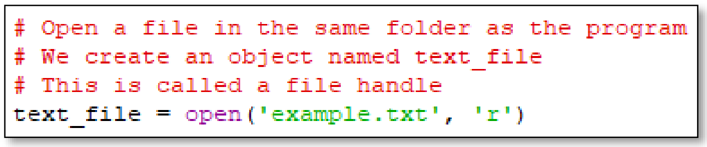
Python File Modes

'r'	Opens a file for reading only. (default)
'w'	Opens a file for writing. Creates a new file if it does not exist or erases the contents of the file if it exists.
'a'	Opens a file for appending at the end of the file without erasing the contents. Creates a new file if it does not exist.

Opening and Reading a Text File

When we want to read or write a file (say on your hard drive), we first must open the file. Opening the file communicates with your operating system, which knows where the data for each file is stored. When you open a file, you are asking the operating system to find the file by name and make sure the file exists.

In this example, we open the file example.txt, which should be stored in the same folder that you are in when you start Python.



```
# Open a file in the same folder as the program
# We create an object named text_file
# This is called a file handle
text_file = open('example.txt', 'r')
```

If the open is successful, the operating system returns us a file handle. The file handle is not the actual data contained in the file, but instead it is a “handle” that we can use to read the data. You are given a handle if the requested file exists and you have the proper permissions to read the file.

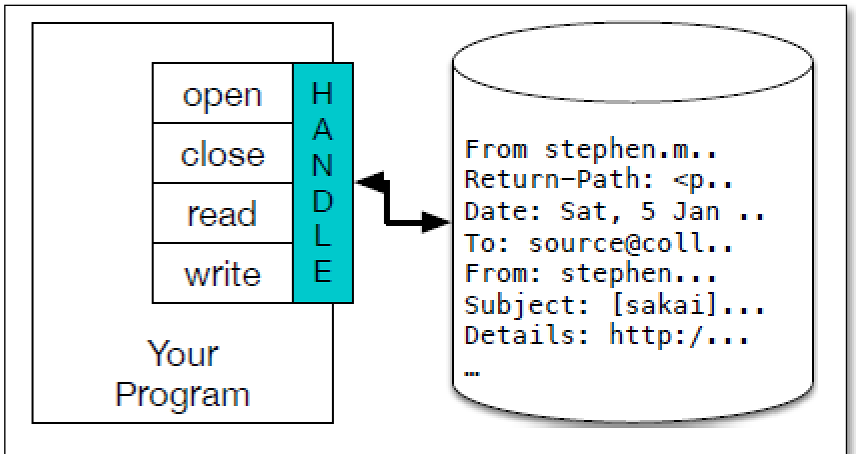


Figure 7.2: A File Handle

The string s is now the following.

```
'Hello.\nThis is a text file.\nBye'
```

If the file does not exist, open will fail with a traceback and you will not get a handle to access the contents of the file.

Directories

Say your program opens a file, like below:

```
s = open('file.txt').read()
```

The file is assumed to be in the same directory as your program itself. If it is in a different directory, then you need to specify the full path to the file, like below:

```
s = open('c:users/loring/desktop/file.txt','r')
```

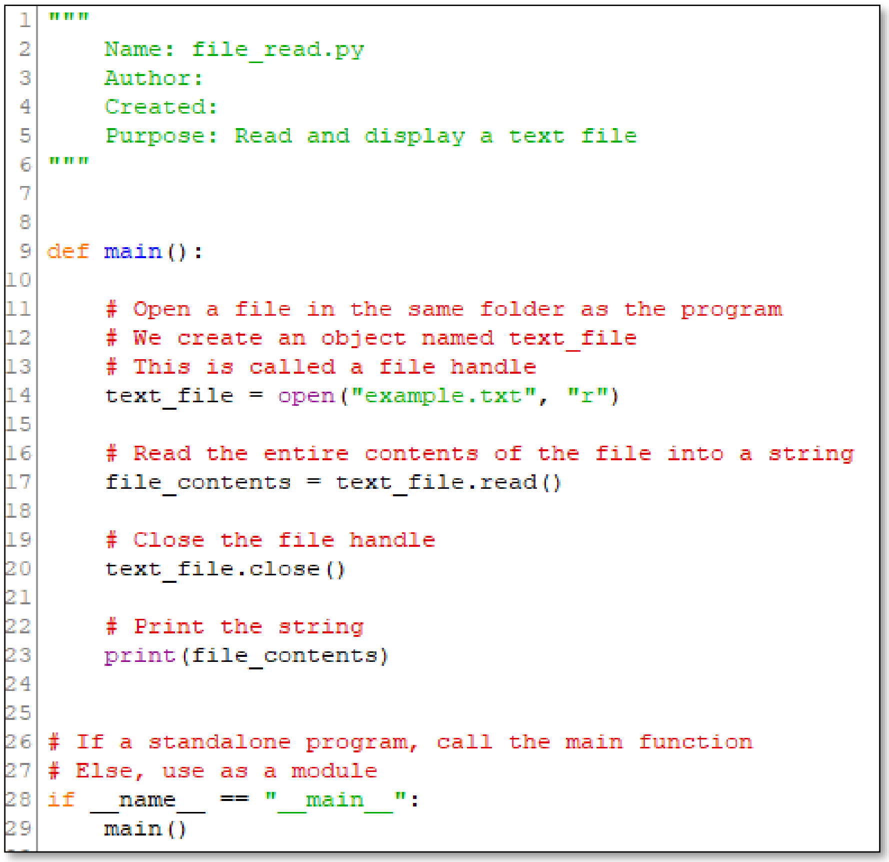
Tutorial 8.1 – Reading a Text File

`'r'`	Opens a file for reading only. (default)

In the same folder, create a txt file named **example.txt** Include the following text.

```
Hello
This is a sample text file.
Bye!
```

Create and test the program as listed. Name it **file_read.py**



```python
"""
    Name: file_read.py
    Author:
    Created:
    Purpose: Read and display a text file
"""


def main():

    # Open a file in the same folder as the program
    # We create an object named text_file
    # This is called a file handle
    text_file = open("example.txt", "r")

    # Read the entire contents of the file into a string
    file_contents = text_file.read()

    # Close the file handle
    text_file.close()

    # Print the string
    print(file_contents)


# If a standalone program, call the main function
# Else, use as a module
if __name__ == "__main__":
    main()
```

Example Working With a File

In this chapter, we explicitly open and close a file. This is to help understand how to work with files. You can also let Python do all the work.

```python
"""
    Name: file_read_rockstars.py
    Author:
    Created:
    Purpose: Read and display a text file using with
"""


def main():
    """
        Open a file in the same folder as the program
        We create an object named file_object
        This is called a file handle
    """
    with open("example.txt") as file_object:

        # Read the entire contents of the file into a string
        file_contents = file_object.read()

    # Print the string
    print(file_contents)


# If a standalone program, call the main function
# Else, use as a module
if __name__ == "__main__":
    main()
```

1. The **with** statement above opens the file with a file handle: **file_object**
2. Inside the **with** statement the file is open. You can perform any file operation inside the with statement.
3. Python closes the file when the program exits the **with** statement.

Writing to a Text File

`'w'`	Opens a file for writing. Creates a new file if it does not exist or erases the contents of the file if it exists.

Let's write to a file called rockstars.txt.

```python
# Create a file handle
# for writing to rockstars.txt
rock_file = open('rockstars.txt', 'w')

# Write the names of three rock stars to the file
# Substitute your favorites
# \n is an escape character creating a new line
rock_file.write(f'Eddie Van Halen\n')
rock_file.write(f'Eric Clapton\n')
rock_file.write(f'Stevie Nicks\n')
```

The first line opens the file rockstars.txt with a file handle named **rock_file**. The 'w' indicates that we want to write to the file.

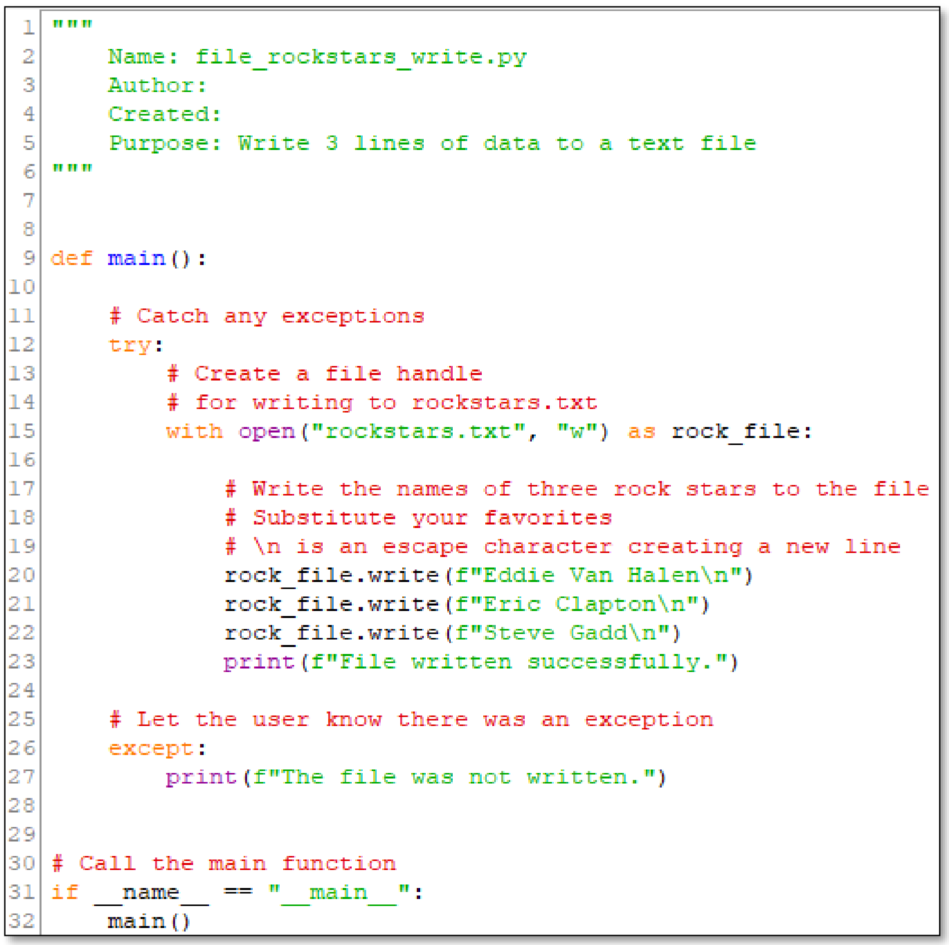
To write to the file, we use the **.write()** method with Fstrings formatting. When we are done writing, we should close the file to make sure our changes take. Be careful here because if writefile.txt already exists, its contents will be overwritten.

Tutorial 8.2 – Writing a Text File

Create and test the following Python program called **file_rockstars_write.py**

This program uses try: except: to provide error handling. If there was an exception, inform the user.

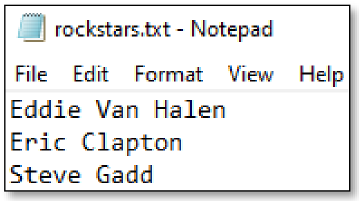
Let the user know the operation was successful by including a print statement in the try block.



```python
"""
    Name: file_rockstars_write.py
    Author:
    Created:
    Purpose: Write 3 lines of data to a text file
"""


def main():

    # Catch any exceptions
    try:
        # Create a file handle
        # for writing to rockstars.txt
        with open("rockstars.txt", "w") as rock_file:

            # Write the names of three rock stars to the file
            # Substitute your favorites
            # \n is an escape character creating a new line
            rock_file.write(f"Eddie Van Halen\n")
            rock_file.write(f"Eric Clapton\n")
            rock_file.write(f"Steve Gadd\n")
            print(f"File written successfully.")

    # Let the user know there was an exception
    except:
        print(f"The file was not written.")


# Call the main function
if __name__ == "__main__":
    main()
```

Open **rockstars.txt** to check your work.

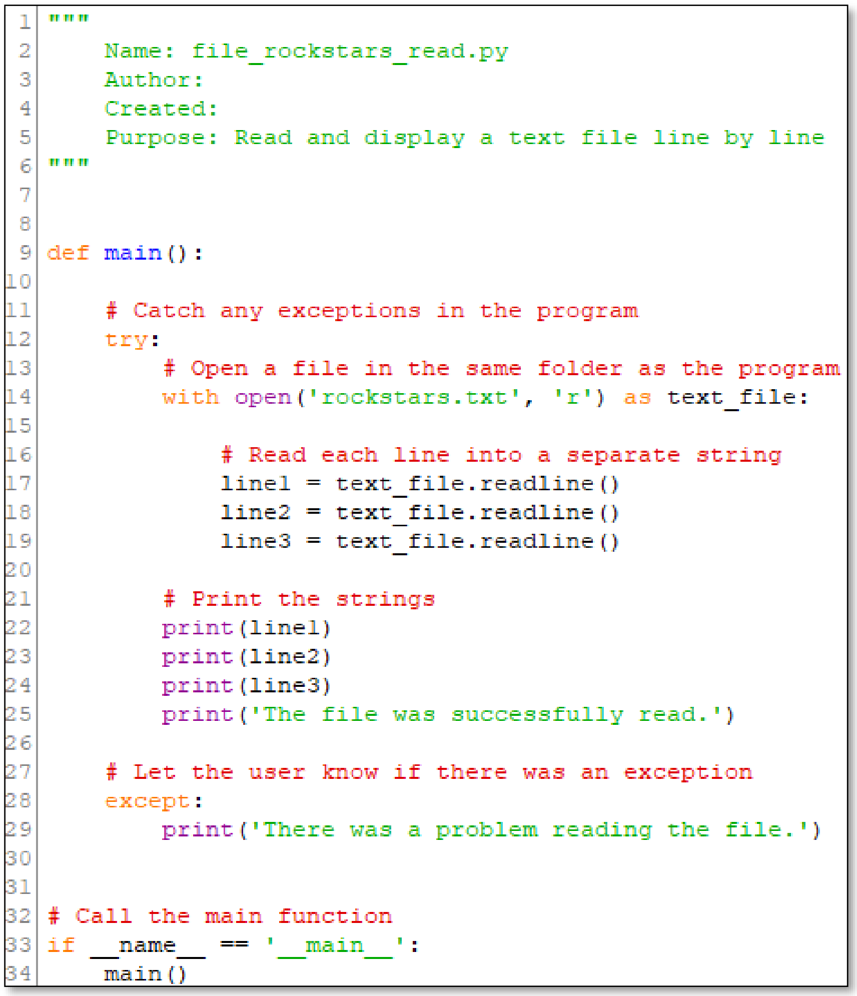
Example run:



```
Eddie Van Halen
Eric Clapton
Steve Gadd
```

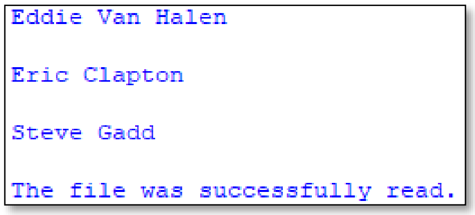
Tutorial 8.3 – Reading a Text File Line by Line

Create a new program called **file_rockstars_read.py** This program reads each line into a separate string.



```python
"""
    Name: file_rockstars_read.py
    Author:
    Created:
    Purpose: Read and display a text file line by line
"""


def main():

    # Catch any exceptions in the program
    try:
        # Open a file in the same folder as the program
        with open('rockstars.txt', 'r') as text_file:

            # Read each line into a separate string
            line1 = text_file.readline()
            line2 = text_file.readline()
            line3 = text_file.readline()

        # Print the strings
        print(line1)
        print(line2)
        print(line3)
        print('The file was successfully read.')

    # Let the user know if there was an exception
    except:
        print('There was a problem reading the file.')


# Call the main function
if __name__ == '__main__':
    main()
```

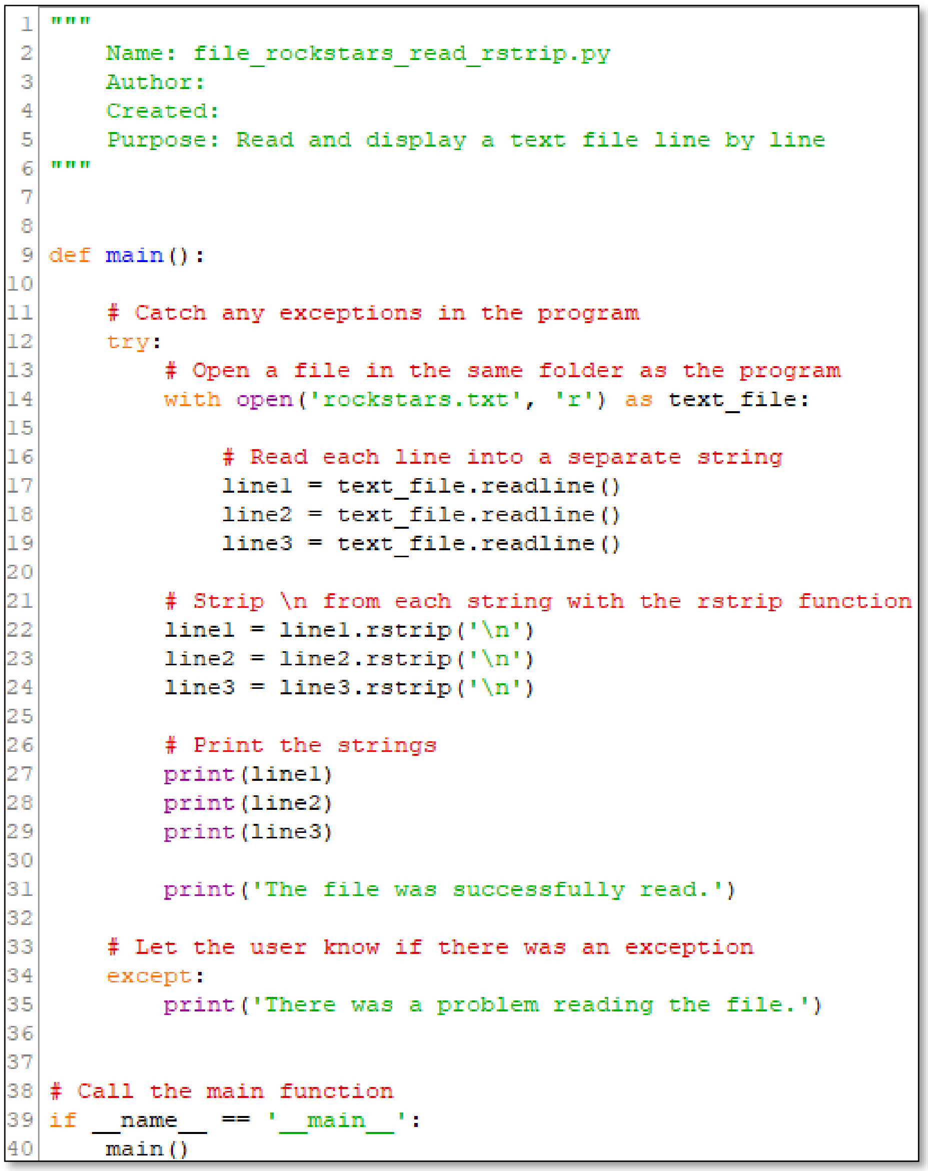
Because of the \n newline character we added when we wrote the file, there is a space between each line when we print out the strings.



```
Eddie Van Halen

Eric Clapton

Steve Gadd

The file was successfully read.
```

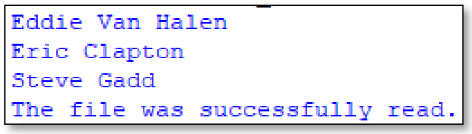
rstrip String Method

We will use the **rstrip()** text method to remove \n from the right side of the string. Add the new code to **file_rockstars_read.py**



```python
"""
    Name: file_rockstars_read_rstrip.py
    Author:
    Created:
    Purpose: Read and display a text file line by line
"""


def main():

    # Catch any exceptions in the program
    try:
        # Open a file in the same folder as the program
        with open('rockstars.txt', 'r') as text_file:

            # Read each line into a separate string
            line1 = text_file.readline()
            line2 = text_file.readline()
            line3 = text_file.readline()

        # Strip \n from each string with the rstrip function
        line1 = line1.rstrip('\n')
        line2 = line2.rstrip('\n')
        line3 = line3.rstrip('\n')

        # Print the strings
        print(line1)
        print(line2)
        print(line3)

        print('The file was successfully read.')

    # Let the user know if there was an exception
    except:
        print('There was a problem reading the file.')


# Call the main function
if __name__ == '__main__':
    main()
```

The text file displayed without \n extra new line characters.

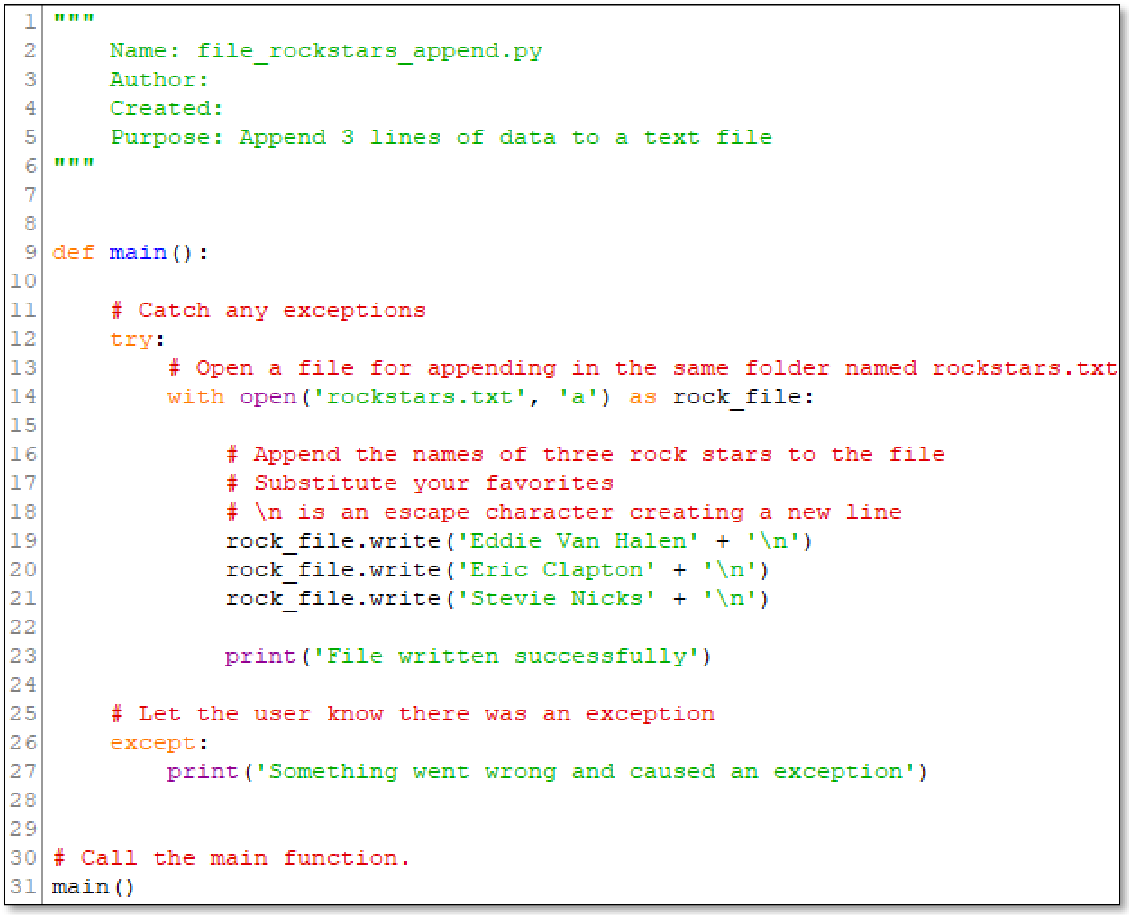


```
Eddie Van Halen
Eric Clapton
Steve Gadd
The file was successfully read.
```

Tutorial 8.4 – Appending to a Text File

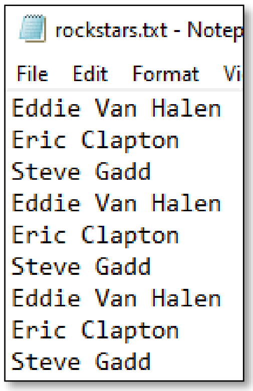
'a'	Opens a file for appending at the end of the file without erasing the contents. Creates a new file if it does not exist.

Open **file_rockstars_write.py**. The only change needed is to change the 'w' to an 'a'. Run the program a couple times. Open the text file to make sure the program worked.



```python
"""
    Name: file_rockstars_append.py
    Author:
    Created:
    Purpose: Append 3 lines of data to a text file
"""


def main():

    # Catch any exceptions
    try:
        # Open a file for appending in the same folder named rockstars.txt
        with open('rockstars.txt', 'a') as rock_file:

            # Append the names of three rock stars to the file
            # Substitute your favorites
            # \n is an escape character creating a new line
            rock_file.write('Eddie Van Halen' + '\n')
            rock_file.write('Eric Clapton' + '\n')
            rock_file.write('Stevie Nicks' + '\n')

            print('File written successfully')

    # Let the user know there was an exception
    except:
        print('Something went wrong and caused an exception')


# Call the main function.
main()
```

Example run:



```
Eddie Van Halen
Eric Clapton
Steve Gadd
Eddie Van Halen
Eric Clapton
Steve Gadd
Eddie Van Halen
Eric Clapton
Steve Gadd
```

File Processing Cheat Sheet

You can **read** an existing file with Python:

```
with open("file.txt") as file:
    content = file.read()
```

You can **create** a new file with Python and **write** some text on it:

```
with open("file.txt", "w") as file:
    content = file.write("Sample text")
```

You can **append** text to an existing file without overwriting it:

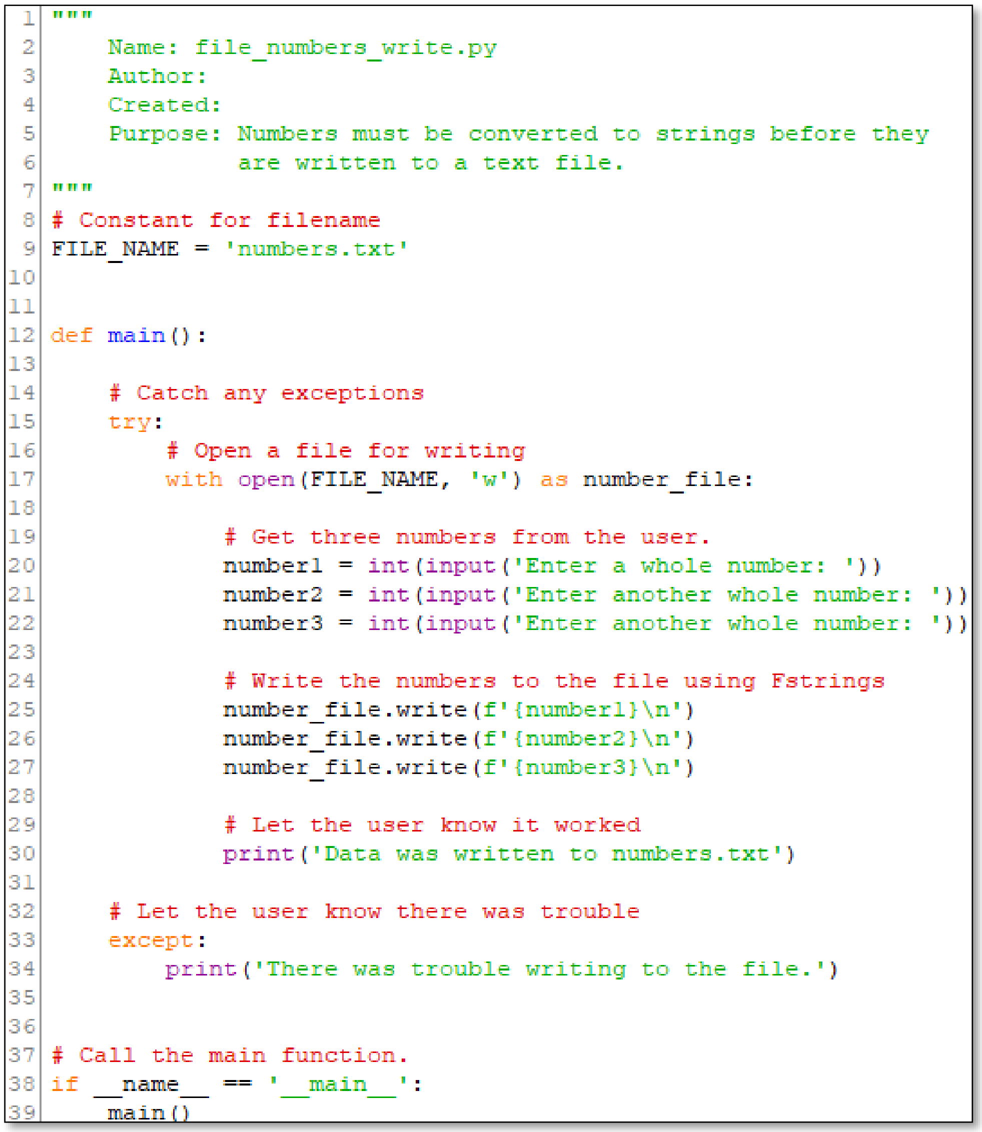
```
with open("file.txt", "a") as file:
    content = file.write("More sample text")
```

You can both **append and read** a file with:

```
with open("file.txt", "a+") as file:
    content = file.write("Even more sample text")
    file.seek(0)
    content = file.read()
```

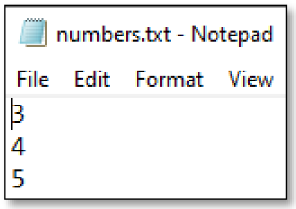
Tutorial 8.5 – Writing Numbers to a Text File

Numbers must be converted to a string before they can be written to a text file. We will use Fstrings to convert the numbers to strings.



```python
"""
    Name: file_numbers_write.py
    Author:
    Created:
    Purpose: Numbers must be converted to strings before they
             are written to a text file.
"""
# Constant for filename
FILE_NAME = 'numbers.txt'


def main():

    # Catch any exceptions
    try:
        # Open a file for writing
        with open(FILE_NAME, 'w') as number_file:

            # Get three numbers from the user.
            number1 = int(input('Enter a whole number: '))
            number2 = int(input('Enter another whole number: '))
            number3 = int(input('Enter another whole number: '))

            # Write the numbers to the file using Fstrings
            number_file.write(f'{number1}\n')
            number_file.write(f'{number2}\n')
            number_file.write(f'{number3}\n')

            # Let the user know it worked
            print('Data was written to numbers.txt')

    # Let the user know there was trouble
    except:
        print('There was trouble writing to the file.')


# Call the main function.
if __name__ == '__main__':
    main()
```

Example run:



```
3
4
5
```

Tutorial 8.6 – Read Numbers from a Text File

Numbers stored as text in a text file must be converted from string to numbers before they can be used in calculations.

```python
"""
    Name: file_read_numbers.py
    Author:
    Created:
    Purpose: Numbers must be converted from strings to ints
"""
# Constant for filename
FILE_NAME = 'numbers.txt'


def main():

    # Catch any exceptions
    try:
        # Open a file for reading
        with open(FILE_NAME, 'r') as number_file:

            # Read 3 numbers from a file
            number1 = int(number_file.readline())
            number2 = int(number_file.readline())
            number3 = int(number_file.readline())

        # Sum the numbers
        total = number1 + number2 + number3

        # Display the numbers and the total
        # A different way of printing numbers as strings
        print(f'The numbers are: {number1} {number2} {number3}')
        print(f'The total is: {total}')

    # Let the user know there was trouble
    except:
        print('There was trouble reading the file.')


# Call the main function.
if __name__ == '__main__':
    main()
```

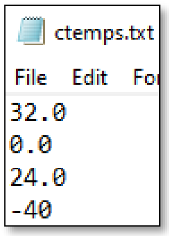
Example run:

```
The numbers are: 3 4 5
The total is: 12
```

Tutorial 8.7 – Reading and Writing to a Text File

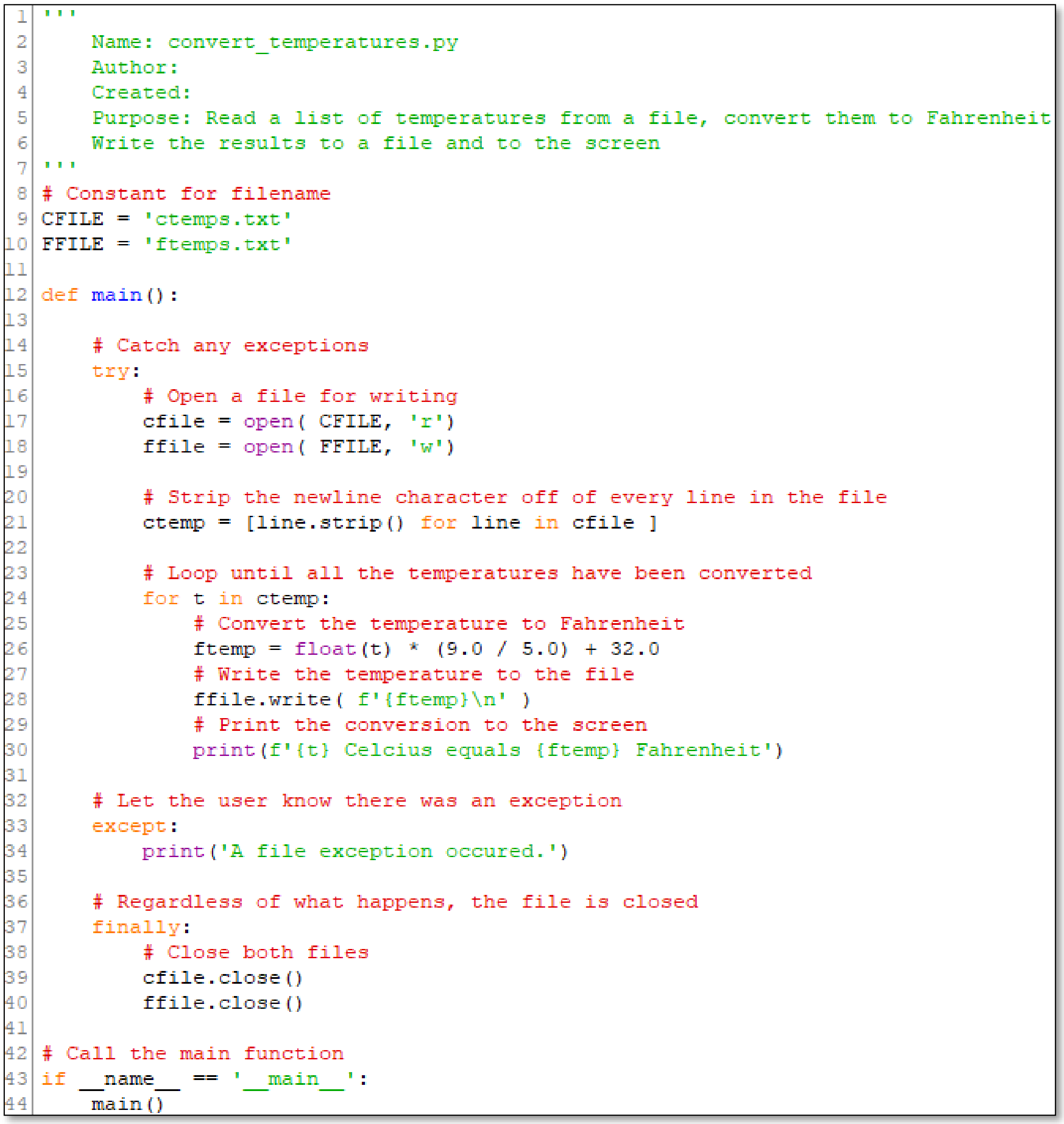
We will write a program that reads a list of temperatures from a file called **ctemps.txt**, converts those temperatures to Fahrenheit, and writes the results to a file called **ftemps.txt**.

Create a text file named **ctemps.txt** in the current folder and put the following numbers in it. Don’t put any extra line returns in it, stop at typing at -40.



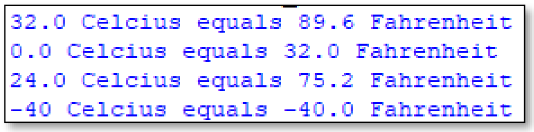
```
32.0
0.0
24.0
-40
```

Create and test the following program named **convert_temperatures.py**

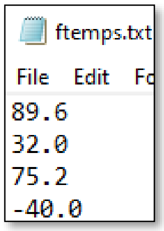


```python
'''
    Name: convert_temperatures.py
    Author:
    Created:
    Purpose: Read a list of temperatures from a file, convert them to Fahrenheit
    Write the results to a file and to the screen
'''
# Constant for filename
CFILE = 'ctemps.txt'
FFILE = 'ftemps.txt'

def main():

    # Catch any exceptions
    try:
        # Open a file for writing
        cfile = open( CFILE, 'r')
        ffile = open( FFILE, 'w')

        # Strip the newline character off of every line in the file
        ctemp = [line.strip() for line in cfile ]

        # Loop until all the temperatures have been converted
        for t in ctemp:
            # Convert the temperature to Fahrenheit
            ftemp = float(t) * (9.0 / 5.0) + 32.0
            # Write the temperature to the file
            ffile.write( f'{ftemp}\n' )
            # Print the conversion to the screen
            print(f'{t} Celcius equals {ftemp} Fahrenheit')

    # Let the user know there was an exception
    except:
        print('A file exception occured.')

    # Regardless of what happens, the file is closed
    finally:
        # Close both files
        cfile.close()
        ffile.close()

# Call the main function
if __name__ == '__main__':
    main()
```

Example run:



```
32.0 Celcius equals 89.6 Fahrenheit
0.0 Celcius equals 32.0 Fahrenheit
24.0 Celcius equals 75.2 Fahrenheit
-40 Celcius equals -40.0 Fahrenheit
```



```
89.6
32.0
75.2
-40.0
```

JSON

Python has a built-in package called **json**, which can be used to work with JSON data.

```
import json
```

JSON (**J**ava**S**cript **O**bject **N**otation) is a popular data format used for representing structured data. It is a common format used to transmit and receive data over the internet or a local network between a server and application.

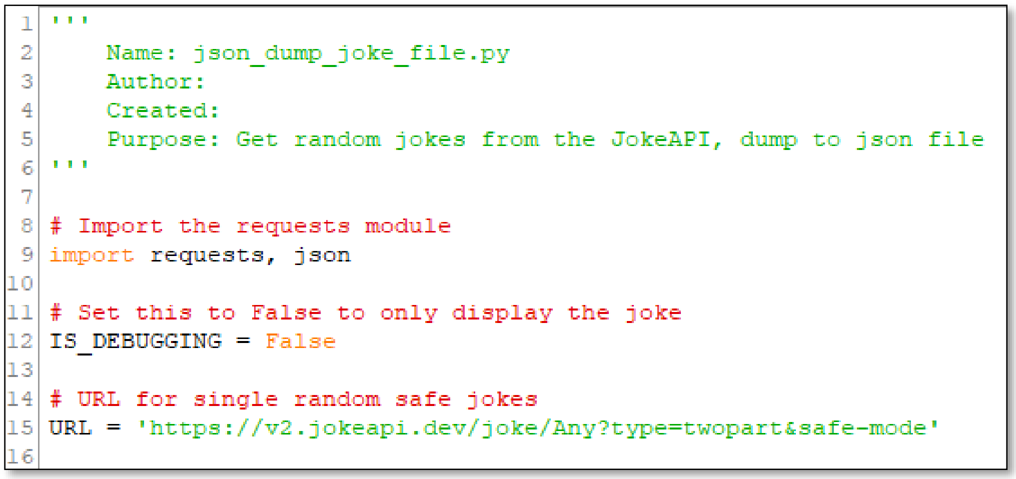
Tutorial 8.8 – Dumping and Loading to a JSON File

Earlier, we created a Python program named **get_joke.py** that retrieved a joke from the Joke API. We are going to save and retrieve the json data to and from a file.

Dump JSON to File

With the json module, there is a **dump** function that dumps data to a json file. We are going to write the Python dictionary to a file. The entire program is included for reference.

Open **get_joke.py** and save it as **json_dump_joke_file.py** Add the indicated code.



```python
'''
    Name: json_dump_joke_file.py
    Author:
    Created:
    Purpose: Get random jokes from the JokeAPI, dump to json file
'''

# Import the requests module
import requests, json

# Set this to False to only display the joke
IS_DEBUGGING = False

# URL for single random safe jokes
URL = 'https://v2.jokeapi.dev/joke/Any?type=twopart&safe-mode'

```

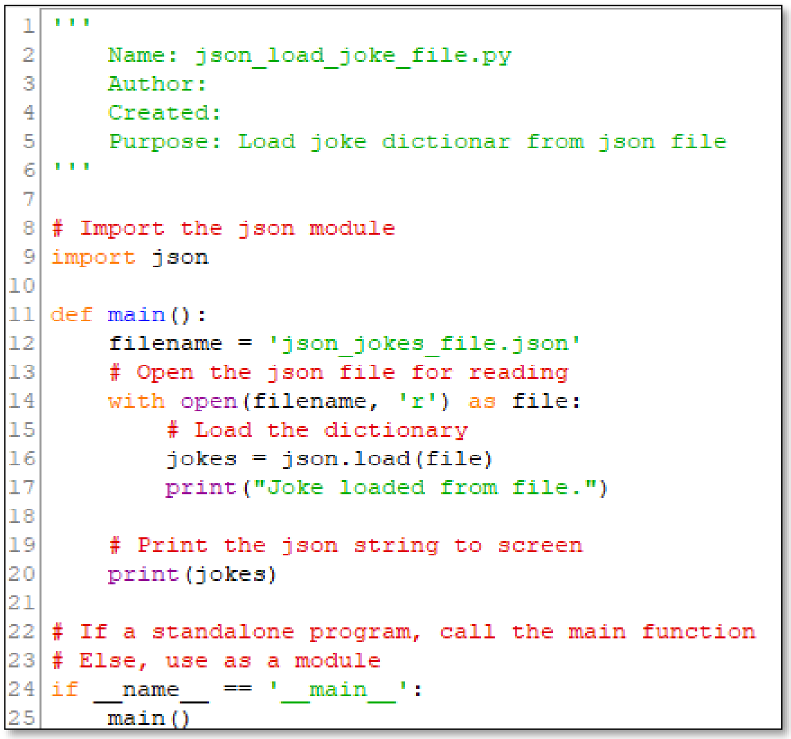
```python
def main():
    filename = 'json_jokes_file.json'

    # Use the requests.get() function
    # with the parameter of the JokeAPI url
    joke = requests.get(URL)

    # If the status_code is 200, successful connection and data
    if(joke.status_code == 200):

        # Convert the JSON data into a Python dictionary with key value pairs
        joke_data = joke.json()

        try:
            with open( filename, 'w' ) as file:
                # Dump the python dictionary to file
                json.dump( joke_data, file )
                print("Joke saved to a file.")
        except:
            print('json data was not dumped to a file.')

        # Used to debug process
        if(IS_DEBUGGING == True):

            # Display the status code
            print(f'\nThe status code for this API request is {joke.status_code}

            # Display the raw JSON data from by Open Notify API
            print('The raw data from the JokeAPI:')
            print(joke.text)

            # Display the Python dictionary
            print('\nThe JSON data converted to a Python dictionary:')
            print(joke_data)

        # Print the data using the dictionary created from the API JSON data
        print('\nTime for a joke!')
        print(f'{joke_data["setup"]}')
        print(f'{joke_data["delivery"]}')

    else:
        print('JokeAPI unavailable')

# If a standalone program, call the main function
# Else, use as a module
if __name__ == '__main__':
    main()
```

Example run:

```
Joke saved to a file.

Time for a joke!
To prove he was right, the flat-earther walked to the end of the Earth.
He eventually came around.
```

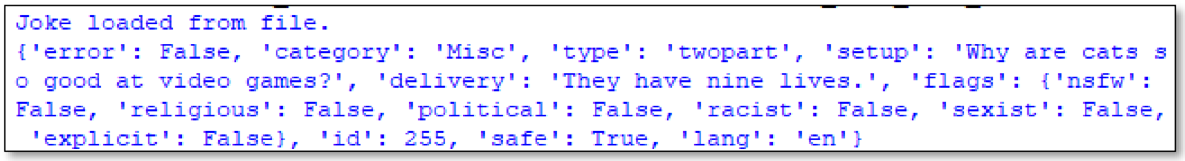
Load JSON Data

The Python json load function loads our dictionary back into memory and displays the data structure.



```python
'''
    Name: json_load_joke_file.py
    Author:
    Created:
    Purpose: Load joke dictionar from json file
'''

# Import the json module
import json

def main():
    filename = 'json_jokes_file.json'
    # Open the json file for reading
    with open(filename, 'r') as file:
        # Load the dictionary
        jokes = json.load(file)
        print("Joke loaded from file.")

    # Print the json string to screen
    print(jokes)

# If a standalone program, call the main function
# Else, use as a module
if __name__ == '__main__':
    main()
```

Example run:



```
Joke loaded from file.
{'error': False, 'category': 'Misc', 'type': 'twopart', 'setup': 'Why are cats s
o good at video games?', 'delivery': 'They have nine lives.', 'flags': {'nsfw':
False, 'religious': False, 'political': False, 'racist': False, 'sexist': False,
 'explicit': False}, 'id': 255, 'safe': True, 'lang': 'en'}
```

Python Pickle

The Python Pickle module is used to perform serialization and deserialization of Python objects. Almost everything in Python is an object, Dictionaries, Lists, Tuples, etc. Unlike saving to a text file, with pickle you can read the object in exactly as it was.

Serializing a Python object means converting it into a byte stream that can be stored in a file or in a string. Pickled data can then be read using the process called deserialization.

Let's start with pickling a list.

Tutorial 8.9 – Pickle and Unpickle a List

Enter following code.

```python
"""
    Name: pickle_list_1.py
    Author:
    Created:
    Purpose: Save and retrieve a Python object
    using the pickle library
"""

import pickle

# A list to pickle
oz_list = ["Lions", "Tigers", "Bears"]
file_name = "oz_list_1.pkl"

print("Original List:")
print(oz_list)

# Pickle the all orders list to a file
# with pickle.dump
with open(file_name, "wb") as fh:
    pickle.dump(oz_list, fh)

print("List dumped to file\n")

# Unpickle the all orders list from a file
# with pickle.load
with open(file_name, "rb") as fh:
    oz_unpickled_list = pickle.load(fh)

print("File loaded to list")
print("Unpickled List:")
print(oz_unpickled_list)
```

Example run:

```
Original List:
['Lions', 'Tigers', 'Bears']
List dumped to file

File loaded to list
Unpickled List:
['Lions', 'Tigers', 'Bears']
```

Tutorial 8.10 – Pickle and Unpickle a Dictionary

Enter the following code:

```python
"""
    Name: pickle_dict_1.py
    Author:
    Created:
    Purpose: Save and retrieve a Python object
    using the pickle library
"""

import pickle

# A Dictionary to pickle
grades_dict = { 'Alex': 87, 'Lini': 92, 'Kiku': 90 }
file_name = "grades_dict_1.pkl"

print("Original Dictionary:")
print(grades_dict)

# Pickle the Dictionary to a file
# with pickle.dump
with open(file_name, "wb") as fh:
    pickle.dump(grades_dict, fh)

print("Dictionary dumped to file\n")

# Unpickle the Dictionary from a file
# with pickle.load
with open(file_name, "rb") as fh:
    grades_unpickled_dict = pickle.load(fh)

print("File loaded to Dictionary")
print("Unpickled Dictionary:")
print(grades_unpickled_dict)
```

Example run:

```
Original Dictionary:
{'Alex': 87, 'Lini': 92, 'Kiku': 90}
Dictionary dumped to file

File loaded to Dictionary
Unpickled Dictionary:
{'Alex': 87, 'Lini': 92, 'Kiku': 90}
```

Glossary

catch To prevent an exception from terminating a program using the try and except statements.

newline A special character used in files and strings to indicate the end of a line.

Pythonic A technique that works elegantly in Python. “Using try and except is the Pythonic way to recover from missing files”.

Quality Assurance A person or team focused on ensuring the overall quality of a software product. QA is often involved in testing a product and identifying problems before the product is released.

text file A sequence of characters stored in permanent storage like a hard drive.

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.